Proof Systems for Retracts in Simply Typed Lambda Calculus

Stirling, C. (2013). Proof Systems for Retracts in Simply Typed Lambda Calculus. In Fomin, F. V., Freivalds, R., Kwiatkowska, M., & Peleg, D. (Eds.), Automata, Languages, and Programming. (pp. 398-409). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. doi: 10.1007/978-3-642-39212-2_36

Proof Systems for Retracts in Simply Typed Lambda Calculus

Colin Stirling

School of Informatics
University of Edinburgh
cps@inf.ed.ac.uk

Abstract. This paper[1] concerns retracts in simply typed lambda calculus assuming $\beta\eta$-equality. We provide a simple tableau proof system which characterises when a type is a retract of another type and which leads to an exponential decision procedure.

1 Introduction

Type ρ is a retract of type τ if there are functions $C : \rho \to \tau$ and $D : \tau \to \rho$ with $D \circ C = \lambda x.x$. This paper concerns retracts in the case of simply typed lambda calculus [1]. Various questions can be asked. The decision problem is: given ρ and τ, is ρ a retract of τ? Is there an independent characterisation of when ρ is a retract of τ? Is there an inductive method, such as a proof system, for deriving assertions of the form "ρ is a retract of τ"? If so, can one also construct (inductively) the witness functions C and D?

Bruce and Longo [2] provide a simple proof system that solves when there are retracts in the case that $D \circ C =_\beta \lambda x.x$. The problem is considerably more difficult if β-equality is replaced with $\beta\eta$-equality. De Liguoro, Piperno and Statman [3] show that the retract relation with respect to $\beta\eta$-equality coincides with the surjection relation: ρ is a retract of τ iff for any model there is a surjection from τ to ρ. They also provide a proof system for the affine case (when each variable in C and D occurs at most once) assuming a single ground type. Regnier and Urzyczyn [9] extend this proof system to cover multiple ground types. The proof systems yield simple inductive nondeterministic algorithms belonging to NP for deciding whether ρ is an affine retract of τ. Schubert [10] shows that the problem of affine retraction is NP-complete and how to derive witnesses C and D from the proof system in [9]. Under the assumption of a single ground type, decidability of when ρ is a retract of τ is shown by Padovani [8] by explicit witness construction (rather than by a proof system) of a special form.

More generally, decidability of the retract problem follows from decidability of higher-order matching in simply typed lambda calculus [13]: ρ is a retract of τ iff the equation $\lambda z^\rho.x_1^{\tau\to\rho}(x_2^{\rho\to\tau}z) =_{\beta\eta} \lambda z^\rho.z$ has a solution (the witnesses D and C for x_1, x_2). Since the complexity of matching is non-elementary [15] this decidability result leaves open whether there is a better algorithm, or even a proof

[1] For a full version see http://www.homepages.inf.ed.ac.uk/cps/ret.pdf

system, for the problem. In the case of β-equality matching is no guide to solvability: the retract problem is simply solvable whereas β-matching is undecidable [4].

In this paper we provide an independent solution to the retract problem. We show it is decidable by exhibiting sound and complete tableau proof systems. We develop two proof systems for retracts, one for the (slightly easier) case when there is a single ground type and the other for when there are multiple ground types. Both proof systems appeal to paths in terms. Their correctness depend on properties of such paths. We appeal to a dialogue game between witnesses of a retract to prove such properties: a similar game-theoretic characterisation of β-reduction underlies decidability of matching.

In Section 2 we introduce retracts in simply typed lambda calculus and fix some notation for terms as trees and for their paths. The two tableau proof systems for retracts are presented in Section 3 where we also briefly examine how they generate a decision procedure for the retract problem. In Section 4 we sketch the proof of soundness of the tableau proof systems (and completeness and further details are provided in the full version).

2 Preliminaries

Simple types are generated from ground types using the binary function operator $\to$. We let $a, b, o, \ldots$ range over ground types and $\rho, \sigma, \tau, \ldots$ range over simple types. Assuming $\to$ associates to the right, so $\rho \to \sigma \to \tau$ is $\rho \to (\sigma \to \tau)$, if a type ρ is not a ground type then it has the form $\rho_1 \to \ldots \to \rho_n \to a$. We say that a is the *target* type of a and of any type $\rho_1 \to \ldots \to \rho_n \to a$.

Simply typed terms in Church style are generated from a countable set of typed variables x^σ using lambda abstraction and function application [1]. We write S^σ, or sometimes $S : \sigma$, to mean term S has type σ. The usual typing rules hold: if S^τ then $\lambda x^\sigma . S^\tau : \sigma \to \tau$; if $S^{\sigma\to\tau}$ and U^σ then $(S^{\sigma\to\tau} U^\sigma) : \tau$. In a sequence of unparenthesised applications we assume that application associates to the left, so $SU_1 \ldots U_k$ is $((\ldots(SU_1)\ldots)U_k)$. Another abbreviation is $\lambda z_1 \ldots z_m$ for $\lambda z_1 \ldots \lambda z_m$. Usual definitions of when a variable occurrence is free or bound and when a term is closed are assumed.

We also assume the usual dynamics of β and η-reductions and the consequent $\beta\eta$-equivalence between terms (as well as α-equivalence). Confluence and strong normalisation ensure that terms reduce to (unique) normal forms. Moreover, we assume the standard notion of η-long β-normal form (a term in normal form which is not an η-reduct of some other term) which we abbreviate to lnf. The syntax of such terms reflects their type: a lnf of type a is a variable x^a, or $x\, U_1 \ldots U_k$ where $x^{\rho_1\to\ldots\to\rho_k\to a}$ and each $U_i^{\rho_i}$ is a lnf; a lnf of type $\rho_1 \to \ldots \to \rho_n \to a$ has the form $\lambda x_1^{\rho_1} \ldots x_n^{\rho_n} . S$, where S^a is a lnf.

The following definition introduces retracts between types [2, 3].

Definition 1. *Type ρ is a* retract *of type τ, written $\models \rho \trianglelefteq \tau$, if there are terms $C : \rho \to \tau$ and $D : \tau \to \rho$ such that $D \circ C =_{\beta\eta} \lambda x^\rho . x$.*

The witnesses C and D to a retract can always be presented as lnfs. We can think of C as a "coder" and D as a "decoder" [9]. Assume $\rho = \rho_1 \to \ldots \to \rho_l \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$: in a retract the types must share target type [9]. We instantiate the bound ρ_i variables in a decoder D to $D(z_1^{\rho_1}, \ldots, z_l^{\rho_l})$, often abbreviated to $D(\overline{z})$, and the bound variable of type ρ in C to $C(x^\rho)$: so, $\models \rho \trianglelefteq \tau$ if $D(z_1^{\rho_1}, \ldots, z_l^{\rho_l})(C(x^\rho)) =_{\beta\eta} xz_1 \ldots z_l$. From [9], we can restrict a decoder to be of the form $\lambda f^\tau . f S_1^{\tau_1} \ldots S_n^{\tau_n}$ with f as head variable and a coder $C(x)$ has the form $\lambda y_1^{\tau_1} \ldots y_n^{\tau_n} . H(xT_1^{\rho_1} \ldots T_l^{\rho_l})$.

Definition 2. *We say that the decoder* $D(z_1, \ldots, z_l) = \lambda f^\tau . f S_1^{\tau_1} \ldots S_n^{\tau_n}$ *and the coder* $C(x) = \lambda y_1^{\tau_1} \ldots y_n^{\tau_n} . H(xT_1^{\rho_1} \ldots T_l^{\rho_l})$ *are* canonical *witnesses for* $\rho \trianglelefteq \tau$ *if* $D(\overline{z})(C(x)) =_{\beta\eta} xz_1 \ldots z_l$ *and they obey the following properties:*

1. *variables* $f, z_1, \ldots, z_l$ *occur only once in* $D(\overline{z})$,
2. x *occurs only once in* $C(x)$,
3. H *is* ε *if* ρ *and* τ *are constructed from a single ground type,*
4. *if* $T_i^{\rho_i}$ *contains an occurrence of* y_j *then it is the head variable of* $T_i^{\rho_i}$, z_i *occurs in* $S_j^{\tau_j}$ *and* $T_i^{\rho_i}$ *contains no other occurrences of any* y_k, $1 \leq k \leq n$.

The next result follows from observations in [3, 9].

Proposition 1. $\models \rho \trianglelefteq \tau$ *iff there exist canonical witnesses for* $\rho \trianglelefteq \tau$.

So, if there is only a single ground type then $C(x)$ can be restricted to have the form $\lambda y_1^{\tau_1} \ldots y_n^{\tau_n} . x T_1^{\rho_1} \ldots T_l^{\rho_l}$ with x as head variable [3].

Example 1. From [3]. Let $\rho = \rho_1 \to \rho_2 \to o$ where $\rho_1 = \rho_2 = \sigma \to o$ and let $\tau = \tau_1 \to o$ where $\tau_1 = \sigma \to (o \to o \to o) \to o$ and σ is arbitrary. It follows that $\models \rho \trianglelefteq \tau$. A decoder $D(z_1^{\rho_1}, z_2^{\rho_2})$ is $\lambda f^\tau . f(\lambda u^\sigma v^{o \to o \to o} . v(z_1 u)(z_2 u))$ and a coder $C(x^\rho)$ is $\lambda y^{\tau_1} . x(\lambda w^\sigma . yw(\lambda s^o t^o . s))(\lambda w^\sigma . yw(\lambda s^o t^o . t))$; so, $(D(z_1, z_2))C(x) \to_\beta^* x(\lambda w^\sigma . z_1 w)(\lambda w^\sigma . z_2 w) =_{\beta\eta} xz_1 z_2$. □

Example 2. From [9] with multiple ground types. Let $\rho = \rho_1 \to \rho_2 \to a$ where $\rho_1 = b \to a$, $\rho_2 = a$ and let $\tau = \tau_1 \to a$ where $\tau_1 = b \to (a \to o \to a) \to a$. A decoder is $D(z_1^{\rho_1}, z_2^{\rho_2})$ is $\lambda f^\tau . f(\lambda u_1^b u_2^{a \to o \to a} . u_2(z_1 u_1) z_2)$ and a coder $C(x^\rho)$ is $\lambda y^{\tau_1} . y s^b (\lambda w_1^a w_2^o . x(\lambda v^b . yv(\lambda w_1^a w_2^o . w_1)) w_2)$; so, $(D(z_1, z_2))C(x) \to_\beta^* x(\lambda v^b . z_1 v) z_2 =_{\beta\eta} xz_1 z_2$. □

Terms are represented as special kinds of tree (that we call *binding* trees in [12, 14]) with dummy lambdas and an explicit binding relation. A term of the form y^a is represented as a tree with a single node labelled y^a. In the case of $y\, U_1 \ldots U_k$, when $y^{\rho_1 \to \cdots \to \rho_k \to a}$, we assume that a dummy lambda with the empty sequence of variables is placed directly above any subterm U_i in its tree representation if ρ_i is a ground type. With this understanding, the tree for $y\, U_1 \ldots U_k$ consists of a root node labelled $y^{\rho_1 \to \cdots \to \rho_k \to a}$ and k-successor trees representing $U_1, \ldots, U_k$. We also use the abbreviation $\lambda \overline{y}$ for $\lambda y_1 \ldots y_m$ for $m \geq 0$, so $\overline{y}$ is possibly the empty sequence of variables in the case of a dummy lambda. The

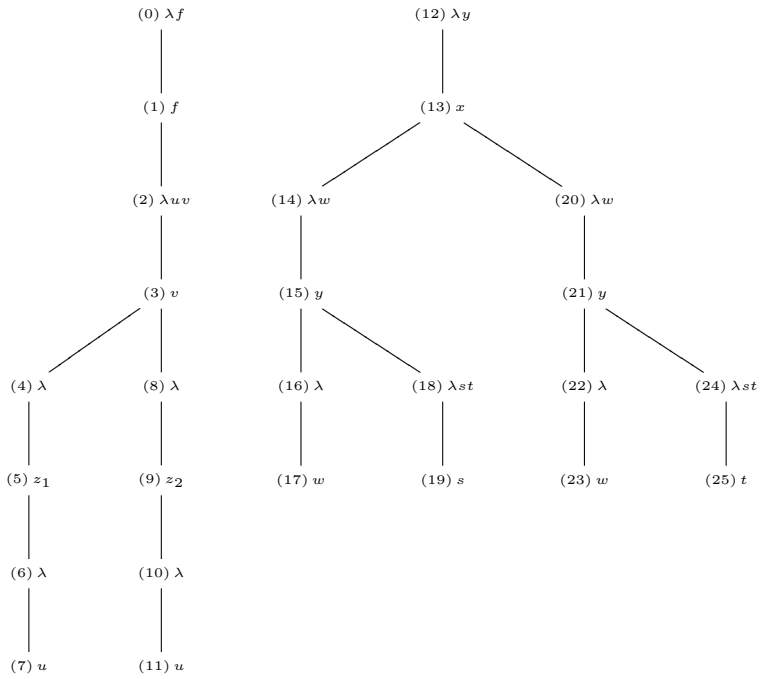

Fig. 1. $D(z_1, z_2)$ and $C(x)$ of Example 1

tree representation of $\lambda\overline{y}.S : \rho_1 \to \ldots \to \rho_k \to a$ consists of a root node labelled $\lambda\overline{y}$ and a single successor tree for S^a. The trees for $C(x)$ and $D(z_1, z_2)$ of Example 1, where we have omitted the types, are in Figure 1.

We say that a node is a *lambda* (*variable*) node if it is labelled with a lambda abstraction (variable). The *type* (*target type*) of a variable node is the type (target type) of the variable at that node and the *type* (*target type*) of a lambda node is the type (target type) of the subterm rooted at that node.

The other elaboration is that we assume an extra binary relation $\downarrow$ between nodes in a tree that represents *binding*; that is, between a node labelled $\lambda y_1 \ldots y_n$ and a node below it labelled y_j (that it binds). A binder $\lambda\overline{y}$ is such that either $\overline{y}$ is empty and therefore is a dummy lambda and cannot bind a variable occurrence or $\overline{y} = y_1 \ldots y_k$ and $\lambda\overline{y}$ can only then bind variable occurrences of the form y_i, $1 \leq i \leq k$. Consequently, we also employ the following abbreviation $n \downarrow_i m$ if $n \downarrow m$ and n is labelled $\lambda y_1 \ldots y_k$ and m is labelled y_i. In Figure 1 we have not included the binding relation; however, for instance, (2) $\downarrow_1$ (7).

Definition 3. *Lambda node n is a* descendant *(*k-descendant*) of m if either $m \downarrow m'$ ($m \downarrow_k m'$), n is a successor of m' for some m' and the target types of m, m' and n are the same, or n' is a descendant (k-descendant) of m and n is a descendant of n' for some n'.*

We assume a standard presentation of nodes of a tree as sequences of integers: an initial sequence, typically ε, is the root node; if n is a node and m is the ith successor of n then $m = ni$. For the sake of brevity we have not followed this approach in Figure 1 where we have presented each node as a unique integer (i).

Definition 4. *A* path *of the tree of a term of type σ is a sequence of nodes $\overline{n} = n_1, \ldots, n_k$ where n_1 is the root of the tree, each n_{i+1} is a successor of n_i and if n_j is a variable node then for some $i < j$, $n_i \downarrow n_j$ (hence is a* closed *path).*

For paths $\overline{m} = m_1, \ldots, m_l$ and $\overline{n} = n_1, \ldots, n_k$ of type σ we write $\overline{m} \sqsubset \overline{n}$ if for some $i > 0$, for all $h \leq 2i$, $m_h = n_h$, $m_{2i+1} = m_{2i}p$, $n_{2i+1} = n_{2i}q$ and $p < q$.

A (closed) subtree *of a tree of a term of type σ is a set of paths P of type σ such that if $\overline{m}, \overline{n}$ are distinct paths in P then $\overline{m} \sqsubset \overline{n}$ or $\overline{n} \sqsubset \overline{m}$.*

A path $\overline{n} = n_1, \ldots, n_k$ is a contiguous sequence of nodes in a tree of a term starting at the root; for $i \geq 1$, each n_{2i-1} is a lambda node and each n_{2i} is a variable node (whose binder occurs earlier in the path). Path $\overline{m}$ is before $\overline{n}$, $\overline{m} \sqsubset \overline{n}$, if they have a common even length prefix and then differ as to their successors (the one in $\overline{m}$ before that in $\overline{n}$). These paths could, therefore, be in the same term: therefore, a closed subtree is a set of such paths.

Definition 5. *A path $\overline{n} = n_1, \ldots, n_l$ is* k-minimal *provided that for each binding node n_i there are at most k distinct nodes n_j, $i < j \leq l$, such that $n_i \downarrow n_j$. A subtree P is* k-minimal *if each path in P is k-minimal.*

Not every path or subtree is useful in a term. So, we define when a path or subtree is realisable meaning that their nodes are "accessible" [7] or "reachable" [6] in an applicative context.

Definition 6. *Assume $\overline{n} = n_1, \ldots, n_l$ is a path of odd length of a closed term T of type σ, m is the node below n_l in T and T' is the term T when the variable u^τ at node m is replaced with a fresh free variable z^τ. We say that $\overline{n}$ is* realisable *if there is a closed term $U = \lambda y^\sigma . y S_1 \ldots S_k$ such that $UT' =_{\beta\eta} \lambda \overline{x}. z W_1 \ldots W_q$ for some $q \geq 0$.*

Definition 7. *Assume P is a subtree of closed term T of type σ where each path has even length, $m_1, \ldots, m_q$ are the leaves of P and T_i, $1 \leq i \leq q$, is the term T when the variable $u_i^{\tau_i}$ at m_i is replaced with a fresh free variable $z_i^{\tau_i}$. We say that P is* realisable *if there is a closed term $U = \lambda y^\sigma . y S_1 \ldots S_k$ such that for each i, $UT_i =_{\beta\eta} \lambda \overline{x}. z_i W_1 \ldots W_{q_i}$ for $q_i \geq 0$.*

Next we define two useful operations on paths, restriction relative to a suffix and the subtype after a prefix.

Definition 8. *Assume that $\overline{n} = n_1, \ldots, n_p$ is a path, $\sigma = \sigma_1 \to \ldots \to \sigma_k \to a$, n_i is a lambda node of type σ and $w = n_i, \ldots, n_p$ is a suffix of $\overline{n}$.*

1. *The suffix w* admits *σ_j, $1 \leq j \leq k$, if either there is no n_q, $i \leq q \leq p$, such that $n_i \downarrow_j n_q$ or there is a j-descendant n_q of n_i whose type is $\tau_1 \to \ldots \to \tau_l \to a$ and for some r there is not a $t : q < t \leq p$ such that $n_q \downarrow_r n_t$ and a is the target type of τ_r.*

2. *The* restriction of σ to w, $\sigma \restriction w$, *is defined as* σ_w *where*
   - $a_w = a$,
   - $(\sigma_j \to \ldots \to \sigma_k \to a)_w =$ *if* w *admits* σ_j *then* $\sigma_j \to (\sigma_{j+1} \to \ldots \to \sigma_k \to a)_w$ *else* $(\sigma_{j+1} \to \ldots \to \sigma_k \to a)_w$.

Definition 9. *Assume that* $\overline{n} = n_1, \ldots, n_p$ *is a path of type* σ. *For a prefix* w *of* $\overline{n}$ *we define the* subtype of σ after w, $w(\sigma)$*:*

- *if* $w = \varepsilon$ *(the empty prefix) then* σ,
- *if* $w = n_1, \ldots, n_q$, $q \leq p$, *then the type of node* n_q.

We also define a canonical presentation of a (prefix or suffix of a) path $\overline{n} = n_1, \ldots, n_k$ of type σ as a *word* w. If w is the empty prefix we write $w = \varepsilon$. Otherwise, $w = (w_1, \ldots, w_j)$, $j \leq k$, where for each $i \geq 0$, $w_{2i+1} = n_{2i+1}$ and if $n_h \downarrow_m n_{2i}$ then $w_{2i} = n_h m$. Thus, we distinguish between $w = \varepsilon$ (the empty word) and $w = (\varepsilon)$ the prefix of length 1 consisting of the root node. Also, we can present a subtree as a set of words. Words will occur in our proof systems as presentations of paths. For example, $w = (\varepsilon, 1, 11, 112, 1112)$ of type τ as in Example 1 represents the path labelled $\lambda f, f, \lambda uv, v, \lambda$ of $D(z_1, z_2)$ in Figure 1 when its root is ε. To illustrate Definitions 8 and 9, for the prefix $w' = (\varepsilon, 1, 11)$ and the suffix $w'' = (11, 112, 1122)$ of w we have $w'(\tau) = \tau_1$ where $\tau_1 = \sigma \to (o \to o \to o) \to o$ as in Example 1 and $\tau_1 \restriction w'' = \sigma \to o$: word w'' of type τ_1 has labelling $\lambda u^{\sigma} v^{o \to o \to o}, v, \lambda$; so, w'' admits the first component σ of τ_1 but not the second $(o \to o \to o)$. The final element of w' is the same as the first element of w''; in such a case we define their concatenation to be w.

Definition 10. *The* concatenation *of (a prefix)* v *and (a suffix)* w, $v^\frown w$, *is:* $\varepsilon^\frown w = w$; *if* $v_k = w_1$ *then* $v_1, \ldots, v_k^\frown w_1, \ldots, w_n = v_1, \ldots, v_k, w_2, \ldots, w_n$.

3 Proof Systems for Retracts

We now develop goal directed tableau proof systems for showing retracts. By inverting the rules one has more classical axiomatic systems: we do it this way because it thereby provides an immediate nondeterministic decision procedure for deciding retracts. We present two such proof systems: a slightly simpler system for the restricted case when there is a single ground type and one for the general case.

3.1 Single Ground Type

Assertions in our proof system are of two kinds. First is $\rho \trianglelefteq \tau$ with meaning ρ is a retract of τ. The second has the form $[\rho_1, \ldots, \rho_k] \trianglelefteq \tau$ which is based on the "product" as defined in [3]. We follow [9] in allowing reordering of components of types since $\rho \to \sigma \to \tau$ is isomorphic to $\sigma \to \rho \to \tau$. Instead we could include explict rules for reordering (as with the axiom in [3]). Moreover, we assume that $[\rho_1, \ldots, \rho_k]$ is a multi-set and so elements can be in any order.

$$I \quad \rho \trianglelefteq \rho$$

$$W \ \frac{\rho \trianglelefteq \sigma \to \tau}{\rho \trianglelefteq \tau}$$

$$C \ \frac{\delta \to \rho \trianglelefteq \sigma \to \tau}{\delta \trianglelefteq \sigma \qquad \rho \trianglelefteq \tau}$$

$$P_1 \ \frac{\rho_1 \to \ldots \to \rho_k \to \rho \trianglelefteq \sigma \to \tau}{[\rho_1, \ldots, \rho_k] \trianglelefteq \sigma \qquad \rho \trianglelefteq \tau}$$

$$P_2 \ \frac{[\rho_1, \ldots, \rho_k] \trianglelefteq \sigma}{\rho_1 \trianglelefteq \sigma \restriction w_1 \quad \ldots \quad \rho_k \trianglelefteq \sigma \restriction w_k} \text{ where}$$

– $w_1 \sqsubset \ldots \sqsubset w_k$ are k-minimal realisable paths of odd length of type σ

Fig. 2. Goal directed proof rules

The proof rules are given in Figure 2. There is a single axiom I, identity, a weakening rule W, a covariance rule C, and two product rules P_1 and P_2. The rules are goal directed: for instance, C allows one to decompose the goal $\delta \to \rho \trianglelefteq \sigma \to \tau$ into the two subgoals $\delta \trianglelefteq \sigma$ and $\rho \trianglelefteq \tau$. I, W and C (or their variants) occur in the proof systems for affine retracts (when variables in witnesses can only occur at most once) [3, 9]. The new rules are the product rules: P_2 appeals to k-minimal realisable paths (presented as words), and the restriction operator of Definition 8. The proof system does not require the axiom A4 of [3], $\sigma \trianglelefteq (\sigma \to a) \to a$: all instances are provable using W and C.

Definition 11. *A* successful *proof tree for $\rho \trianglelefteq \tau$ is a finite tree whose root is labelled with the goal $\rho \trianglelefteq \tau$, the successor nodes of a node are the result of an application of one of the rules to it, and each leaf is labelled with an axiom. We write $\vdash \rho \trianglelefteq \tau$ if there is a successful proof tree for $\rho \trianglelefteq \tau$.*

For some intuition about the product rules assume $\rho = \rho_1 \to \ldots \to \rho_l \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$. Now, $\models \rho \trianglelefteq \tau$ iff there are canonical, Definition 2, witnesses $D(z_1^{\rho_1}, \ldots, z_l^{\rho_l}) = \lambda f^{\tau}.f S_1^{\tau_1} \ldots S_n^{\tau_n}$. Since we can reorder components of ρ and τ we can assume that z_1 is in $S_1^{\tau_1}$. Suppose $z_1, \ldots, z_k$, where $k > 1$, are in $S_1^{\tau_1}$ and so y_1 must occur in $T_1^{\rho_1}, \ldots, T_k^{\rho_k}$. Therefore, there is a common coder $S_1^{\tau_1}(x_1/z_1, \ldots, x_k/z_k)$ and k decoders $T_i(\overline{z}_i)$ where $\overline{z}_i = z_{i1}^{\rho_{i1}}, \ldots, z_{il_i}^{\rho_{il_i}}$ and $\rho_{i1}, \ldots, \rho_{il_i}$ are the components of ρ_i such that $T_i(\overline{z}_i)(S_1^{\tau_1}(x_1, \ldots, x_k)) =_{\beta\eta} x_i \overline{z}_i$ (which is similar to the product in [3]). In $S_1^{\tau_1}(x_1/z_1, \ldots, x_k/z_k)$ there are distinct odd length paths $w_1, \ldots, w_k$ of type τ_1 to the lambda nodes above $x_1, \ldots, x_k$. These paths are realisable, Definition 6, because each x_i belongs to the normal form of $T_i(\overline{z}_i)(S_1^{\tau_1}(x_1, \ldots, x_k))$. Using a combinatorial argument, see the full version, $S_1^{\tau_1}$ can be chosen so that these words are k-minimal and

$$\dfrac{(\sigma \to o) \to (\sigma \to o) \to o \trianglelefteq (\sigma \to (o \to o \to o) \to o) \to o}{\dfrac{[\sigma \to o, \sigma \to o] \trianglelefteq \sigma \to (o \to o \to o) \to o \quad o \trianglelefteq o}{\sigma \to o \trianglelefteq \sigma \to o \quad \sigma \to o \trianglelefteq \sigma \to o}}$$

Fig. 3. A proof tree for Example 1

by reordering ρ's components $w_1 \sqsubset \ldots \sqsubset w_k$. We may not be able to reduce to the subgoals $\rho_1 \trianglelefteq \tau_1, \ldots, \rho_k \trianglelefteq \tau_1$ as w_i may prescribe the form of $T_i(\overline{z}_i)$: if $T_i(\overline{z}_i) = \lambda f^{\tau_1}.f S_1^i \ldots S_m^i$ then path w_i may prevent S_j^i containing elements of $\overline{z}_i$; so, this may restrict the possible distribution of $\overline{z}_i$ within the subterms $S_1^i, \ldots, S_m^i$ which is captured using $\tau_1 \upharpoonright w_i$.

An example proof tree is in Figure 3 for the retract of Example 1 (which is not affine). Rule P_1 is applied to the root and then P_2 to the first subgoal where $w_1 = (\varepsilon, 2, 21)$ and $w_2 = (\varepsilon, 2, 22)$. Let $\sigma' = \sigma \to (o \to o \to o) \to o$. Now, $\sigma' \upharpoonright w_1 = \sigma \to o = \sigma' \upharpoonright w_2$; in both cases only the first component of σ' is admitted.

3.2 Multiple Ground Types

We extend the proof system to include multiple ground types. Again, assertions are of the two kinds $\rho \trianglelefteq \tau$ and $[\rho_1, \ldots, \rho_k] \trianglelefteq \tau$. However, we now assume that to be a well-formed assertion $\rho \trianglelefteq \tau$ both ρ and τ must share the same target type (which is guaranteed when there is a single ground type). The rules for this assertion are as before the axiom I, weakening W, covariance C and the product rule P_1 in Figure 2: however, C carries the requirement that the target types of δ and σ coincide. The other product rule P_2', presented in Figure 4, is different: the *arity* of $\rho_1 \to \ldots \to \rho_n \to a$ is the maximum of n and the arities of each ρ_i where a gound type a has arity 0.

$$P_2' \ \dfrac{[\rho_1, \ldots, \rho_k] \trianglelefteq \sigma}{\rho_1 \trianglelefteq v_1(\sigma) \upharpoonright w_1 \quad \ldots \quad \rho_k \trianglelefteq v_k(\sigma) \upharpoonright w_k} \text{ where}$$

- k' is the maximum of k and h^2 where h is the arity of σ
- there is a k'-minimal realisable subtree U of type σ where each path has even length (which can be $\emptyset$),
- each v_i is ε, a prefix of a path in U of odd length or the extension of a path in U with a single node,
- $v_1^\frown w_1 \sqsubset \ldots \sqsubset v_k^\frown w_k$ and each $v_i^\frown w_i$ is a k'-minimal realisable path of type σ of odd length and if $U \neq \emptyset$, $v_i^\frown w_i$ extends some path in U.

Fig. 4. Product proof rule for multiple gound types

$$\dfrac{(b \to a) \to o \to a \trianglelefteq (b \to (a \to o \to a) \to a) \to a}{\dfrac{[b \to a, o] \trianglelefteq b \to (a \to o \to a) \to a \quad a \trianglelefteq a}{b \to a \trianglelefteq b \to a \quad o \trianglelefteq o}}$$

Fig. 5. A proof tree for Example 2

In $[\rho_1, \ldots, \rho_k] \trianglelefteq \sigma$ it is not required that ρ_j and σ share the same target type. Instead rule P_2' requires that ρ_i and $v_i(\sigma)$, see Definition 9, do share target types: for the concatenation $v_i^\wedge w_i$ see Definition 10. The specialisation to the case of the single ground type is when $U = \emptyset$ and $v = \varepsilon$.

Let $\rho = \rho_1 \to \ldots \to \rho_l \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$. So, $\models \rho \trianglelefteq \tau$ iff there are canonical witnesses $D(z_1^{\rho_1}, \ldots, z_l^{\rho_l}) = \lambda f^\tau . f S_1^{\tau_1} \ldots S_n^{\tau_n}$ and $C(x) = \lambda y_1^{\tau_1} \ldots y_n^{\tau_n} . H(x T_1^{\rho_1} \ldots T_l^{\rho_l})$. Assume $z_1, \ldots, z_k$, where $k \geq 1$, occur in $S_1^{\tau_1}$. There is a path v in $C(x)$ to the node above x which determines a subtree U of $S_1^{\tau_1}$. The head variable in $T_i^{\rho_i}$ bound in v has the same target type as ρ_i. There are distinct paths $v_1^\wedge w_1, \ldots, v_k^\wedge w_k$ of odd length to the lambda nodes above $z_1, \ldots, z_k$ in $S_1^{\tau_1}$: v_i is decided by the meaning of the head variable in $T_i^{\rho_i}$; so, $v_i(\tau_1)$ has the same target type as ρ_i. The rest of the path is the tail of w_i: so we need to consider whether $\models \rho_i \trianglelefteq v_i(\tau_1) \upharpoonright w_i$.

Figure 5 is the proof tree for the retract in Example 2. There is an application of P_1 followed by P_2'. In the application of P_2' the subtree $U = \{(\varepsilon, 2)\}$, $v_1 = \varepsilon$, $w_1 = (\varepsilon, 2, 21) = v_1^\wedge w_1$, $v_2 = (\varepsilon, 2, 22) = v_2^\wedge w_2$ when $w_2 = (22)$. So, $v_1(b \to (a \to o \to a) \to a) \upharpoonright w_1 = b \to a$ as the first component is admitted (unlike the second); and $v_2(b \to (a \to o \to a) \to a) = o = o \upharpoonright w_2$.

3.3 Complexity

The proof systems provide nondeterministic decision procedures for checking retracts. Each subgoal of a proof rule has smaller size than the goal. Hence, by focussing on one subgoal at a time a proof witness can be presented in PSPACE. However, this does not take into account checking that a subgoal obeys the side conditions in the case of the product rules. Given any type σ, there are boundedly many realisable k-minimal paths (with an upper bound of k^n where n is size of σ). So, this means that overall the decision procedure requires at most EXPSPACE.

4 Soundness and Completeness

To show soundness and completeness of our proof systems, we define a dialogue game $\mathsf{G}(D(\overline{z}), C(x))$ played by a single player $\forall$ on the trees of potential witnesses for a retract that characterises when $(D(\overline{z}))C(x) =_{\beta\eta} x\overline{z}$, similar to game semantics [5]. The game is defined in the full version of the paper.

To provide intuition for the reader we briefly describe $\mathsf{G}(D(z_1, z_2), C(x))$ where these terms are from Figure 1. Play starts at node (0), the binder λf at that node is associated with $C(x)$ rooted at (12); so, the next position is at node (1) and therefore jumps to (12); the binder at (12) λy is associated with node (2) (the successor of (1)). Play proceeds to (13) and $\forall$ chooses to go left or right; suppose it is left, so play is then at (14); nodes (13) and (14) are part of the normal form, see Definition **??**. Play descends to (15) and, therefore, jumps to (2); so, with the binder at (2), u is associated with the the subtree at (16) and v with the subtree at (18). Play proceeds to (3) and so jumps to (18); now, s is associated with (4) and t with (8). Play proceeds to (19) and so jumps to (4), descends to (5) and then to (6) and then to (7) and jumps to (16) before finishing at (17). This play captures the path $x\lambda w.z_1 w$ of the normal form.

Some of the key properties, defined in the full version, we appeal to in the correctness proofs below *associate* subtrees with realisable paths and vice versa. For instance, as illustrated in the play above the path rooted at (0) downto (7) is associated with the subtree rooted at (12) and with leaves (17) and (19). Let $\rho = \rho_1 \to \ldots \to \rho_l \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$ and let $\tau_1 = \sigma = \sigma_1 \to \ldots \to \sigma_m \to b$.

Theorem 1. *(Soundness) If* $\vdash \rho \trianglelefteq \tau$ *then* $\models \rho \trianglelefteq \tau$.

Proof. By induction on the depth of a proof. For the base case, the result is clear for a proof that uses the axiom I. So, assume the result for all proofs of depth $< d$. Consider now a proof of depth d. We proceed by examining the first rule that is applied to show $\vdash \rho \trianglelefteq \tau$. If it is W or C the result follows using the same arguments as in [3]. Assume the rule is W and suppose $\models \rho \trianglelefteq \tau$. Therefore there are terms D_1 and C_1 such that $D_1^{\tau\to\rho}(C_1^{\rho\to\tau x}) =_{\beta\eta} x$. Now $D^{(\sigma\to\tau)\to\rho} = \lambda f^{\sigma\to\tau} y^{\sigma}.D_1(fy)$ and $C^{\rho\to(\sigma\to\tau)}x = \lambda s^{\sigma}.C_1(x)$ are witnesses for $\models \rho \trianglelefteq \sigma \to \tau$. Assume that the rule is C, so $\models \delta \trianglelefteq \sigma$ and $\models \rho \trianglelefteq \tau$. So there are terms D_1, C_1, D_2, C_2 such that $D_1^{\sigma\to\delta}(C_1^{\delta\to\sigma}x) =_{\beta\eta} x$ and $D_2^{\tau\to\rho}(C_2^{\rho\to\tau}x) =_{\beta\eta} x$. Now $D^{(\sigma\to\tau)\to(\delta\to\rho)} = \lambda xy.C_2(x(D_1 y))$ and $C^{(\delta\to\rho)\to(\sigma\to\tau)} = \lambda uz.D_2(u(C_1 z))$ are witnesses for $\models \delta \to \rho \trianglelefteq \sigma \to \tau$.

Consider next that the first rule is P_1. So after P_1 there is either an application of P_2 or P_2': in the former case, there are k-minimal realisable paths $w_1 \sqsubset \ldots \sqsubset w_k$ of odd length of type σ such that $\vdash \rho_i \trianglelefteq \sigma \upharpoonright w_i$; in the latter case, there is a k'-minimal realisable subtree U of type σ where each path has even length; and there are paths $v_1^\frown w_1 \sqsubset \ldots \sqsubset v_k^\frown w_k$ where each element is a k'-minimal realisable path of type σ of odd length and if $U \neq \emptyset$, it extends some path in U and where each v_i is ε, a prefix of a path in U of odd length path or an extension of a path in U with a single node and $\vdash \rho_i \trianglelefteq v_i(\sigma) \upharpoonright w_i$; where k' is the maximum of k and the square of the arity of σ. So, by the induction hypothesis there are terms $D_i(\overline{z}_i)$ and $C_i(x_i)$ such that $D_i(\overline{z}_i)(C_i(x_i)) =_{\beta\eta} x_i\overline{z}_i$, witnesses for $\rho_i \trianglelefteq \sigma \upharpoonright w_i$ or $\rho_i \trianglelefteq v_i(\sigma) \upharpoonright w_i$, and terms $D'(z_{k+1}, \ldots, z_l)$ and $C'(x')$ such that $D'(z_{k+1}, \ldots, z_l)(C'(x')) =_{\beta\eta} x' z_{k+1} \ldots z_l$, witnesses for $\rho_{k+1} \to \ldots \to \rho_l \to a \trianglelefteq \tau'$ where $\tau' = \tau_2 \to \ldots \to \tau_n \to a$. We assume that all these terms are canonical witnesses. The term $D'(z_{k+1}, \ldots, z_l)$

is $\lambda f^{\tau'}.fS_2^{\tau_2}\dots S_n^{\tau_n}$ and $C'(x')$ is $\lambda y_2^{\tau_2}\dots y_n^{\tau_n}.H'(x'T_{k+1}^{\rho_{k+1}}\dots T_l^{\rho_l})$ where $H'=\varepsilon$ if the rule applied was P_2.

We need to show that there are terms $D(z_1,\dots,z_l)$ and $C(x)$ that are witnesses for $\models \rho \trianglelefteq \tau$. $D(\overline{z})$ will have the form $\lambda f^{\tau}.fS_1^{\tau_1}\dots S_n^{\tau_n}$ and $C(x)$ the form $\lambda y_1^{\tau_1}\dots y_n^{\tau_n}.H(xT_1^{\rho_1}\dots T_l^{\rho_l})$ where $H=\varepsilon$ in the case of a single ground type. All that remains is to define $S_1^{\tau_1}$ so it contains $z_1,\dots,z_k$, $T_1^{\rho_1},\dots,T_k^{\rho_k}$ and H (as an extension of H'). If $U=\emptyset$ then $H=H'$. Otherwise, let u be an odd length path such that U is associated with (so, its head variable is $y_1^{\tau_1}$). H consists of the suffix of u followed by the subtree H'. The head variable of each $T_i^{\rho_i}$ is y_1 in the case of the single ground type and $g_i^{v_i(\sigma)}$ in the general case (which is either y_1 or bound in u). We assume that S'_i is the subterm of S_1^{σ} that is rooted at the initial vertex of the path w_i: which is S_1^{σ} itself in the single ground type. To complete these terms we require that $T_i^{\rho_i}(S_1^{\sigma}(z_1,\dots,z_k)) =_{\beta\eta} z_i$. Therefore, removing lambda abstraction over variables z_{ij} and changing z_i to x_i, we require that $T_i(\overline{z}_i)(S'_i(x_1,\dots,x_k)) =_{\beta\eta} x_i\overline{z}_i$. We construct a term $C''(x_i)$ that occurs after the path w_i in S'_i (and which has root x_i when there is a single ground type). We also complete $T_i(\overline{z}_i)$ whose initial part is the tree U_i associated with the path w_i.

First, we examine the single ground type case. So, S_1^{σ} will have the form $\lambda u_1\dots u_m.S'_1$, $C''(x_i)$ the form $x_iC''_{i1}\dots C''_{ip}$ and $T_i(\overline{z}_i)$ the form $\lambda f_i^{\sigma}.f_iV_1^i\dots V_m^i$. Assume $D_i(\overline{z}_i)$ is $\lambda g_i^{\sigma\restriction w_i}.g_iW_{i_1}^i\dots W_{i_l}^i$ and $C_i(x_i)$ is $\lambda u_{i_1}\dots u_{i_l}.x_iC_1^i\dots C_p^i$. Assume w_i admits σ_{i_j}: therefore, for some $r: 1\le r\le m$, $i_j=r$ (so, W_r^i may contain occurrences of variables in $\overline{z}_i$). If u_r does not occur in the path w_i then we set $V_r^i=W_r^i$. Otherwise, there is a non-empty subpath w_{ir} of w_i generated by u_r, and a subtree U_r^i of V_r^i associated with w_{ir}. Each C_j^i contains a single u_{i_k} (as head variable). Assume C_s^i contains u_r. Assume that the path in W_r^i to the lambda node above z_{is} is w'_s. If we can build the same path in V_r^i (by copying nodes of C_s^i to C''_{is}) then we are done (letting V_r^i include this path followed by the subterm of W_r^i rooted at z_{is}). Otherwise, we initially include w_{ir} in C''_{is} and then try to build w'_s in V_r^i by copying nodes of C_s^i to C''_{is}: in V_r^i and, therefore in U_r^i, there is a path whose prefix except for its final variable vertex is the same as a prefix of w'_s and then differ. In the game $\mathsf{G}(C''_{is},V_r^i)$, play jumps from that variable in V_r^i to a lambda node in w_{ir}. By definition of admits, there is a binder n' labelled $\lambda\overline{v}$ in w_{ir} such that for some q not$(n'\downarrow_q n'_i)$ for all nodes n'_i after n' in w_i (and in w_{ir}). Therefore, we add a variable node labelled v_q to the end of w_{ir} in C''_{is}; so play jumps to a lambda node in V_r^i which is a successor of a leaf of U_r^i; below this node, we build the path w'_s except for its root node (by adding further nodes to C''_{is} and add the subtree rooted at z_{is} in W_r^i to V_r^i).

For the general case, assume $v_i(\sigma)=\sigma'_1\to\dots\to\sigma'_m\to b$. So, $S_i^{v_i(\sigma)}$ will have the form $\lambda u_1\dots u_m.S'_1$, $C''(x_i)$ the form $H_i(x_iC''_{i1}\dots C''_{ip})$ and $T_i(\overline{z}_i)$ the form $\lambda f_i^{\sigma}.f_iV_1^i\dots V_m^i$. Assume $D_i(\overline{z}_i)$ is $\lambda g_i^{v_i(\sigma)\restriction w_i}.g_iW_{i_1}^i\dots W_{i_l}^i$ and $C_i(x_i)$ is $\lambda u'_{i_1}\dots u'_{i_l}.H'_i(x_iC_1^i\dots C_p^i)$. We set $H_i=H'_i$. Then we proceed in a similar fashion to the single base type case. If some u'_r does not occur in the path w_i then

$V_r^i = W_r^i$; otherwise we need to build similar paths to z_{is} in W_r^i in V_r^i (by copying vertices from C_s^i to C_{is}'' and using that w_i admits $(v_i(\sigma))_r$. □

The proof of completeness (by induction on the size of ρ) is easier.

Theorem 2. *(Completeness) If* $\models \rho \trianglelefteq \tau$ *then* $\vdash \rho \trianglelefteq \tau$.

5 Conclusion

We have provided tableau proof systems that characterise when a type is a retract of another type in simply typed lambda calculus (with respect to $\beta\eta$-equality). They offer a a nondeterministic decision procedure for the retract problem in EXPSPACE: it may be possible to improve on the rather crude k-minimality bounds used on paths within the proof systems. Given the constructive proof of correctness, we also expect to be able to extract witnesses for a retract from a successful tableau proof tree (similar in spirit to [10]).

References

1. Barendregt, H. Lambda calculi with types. In *Handbook of Logic in Computer Science, Vol 2*, ed. Abramsky, S., Gabbay, D. and Maibaum, T., Oxford University Press, 118-309, (1992).
2. Bruce, K. and Longo, G. Provable isomorphisms and domain equations in models of typed languages. *Proc. 17th Symposium on Theory of Computing*, ACM, 263-272, (1985).
3. de'Liguoro, U., Piperno, A. and Statman, R. Retracts in simply typed $\lambda\beta\eta$-calculus. *Procs. LICS 1992*, 461-469, (1992).
4. Loader, R. Higher-order β-matching is undecidable, *Logic Journal of the IGPL*, 11(1), 51-68, (2003).
5. Ong, C.-H. L. On model-checking trees generated by higher-order recursion schemes, *Procs LICS 2006*, 81-90.
6. Ong and Tzevelekos. Functional Reachability. *Procs LICS 2009*, 286-295, (2009).
7. Padovani, V. Decidability of fourth-order matching. *Mathematical Structures in Computer Science*, 10(3), 361-372, (2000).
8. Padovani, V. Retracts in simple types. In: Abramsky, S. (ed.) TLCA 2001. *LNCS*, 2044, 376-384, (2001)
9. Regnier, L. and Urzyczyn, P. Retractions of types with many atoms. At http://arxiv.org/abs/cs/0212005, pp1-16.
10. Schubert, A. On the building of affine retractions. *Math. Struct. in Comp. Science*, 18, 753-793, (2008).
11. Stirling, C. Higher-order matching, games and automata. *Procs LICS 2007*, 326-335, (2007).
12. Stirling, C. Dependency tree automata. In: de Alfaro, L. (ed.) FOSSACS 2009. *LNCS*, 5504, 92-106, (2009).
13. Stirling, C. Decidability of higher-order matching *Logical Methods in Computer Science*, 5(3:2), 1-52, (2009).
14. Stirling, C. An introduction to decidability of higher-order matching. *Submitted for publication.* Available at author's website, (2012).
15. Vorobyov, S. The "hardest" natural decidable theory. *Procs LICS 1997*, 294-305, (1997).